Explaining Query Answers in Lightweight Ontologies:
The *DL-Lite* Case

Giorgio Stefanoni  
Supervisor: Prof. T. Eiter  
Co-Supervisors: Dr. M. Ortiz  Dr. M. Šimkus  
Scientific Advisor: D. Calvanese

Department of Computer Science  
University of Oxford, UK

February 20, 2012
Outline

1. Foundations
2. Explaining Positive Answers
3. Explaining Negative Answers
4. Conclusions
Query Answering in Description Logics

Logical Reasoning

Query $q$
Ontology $T$
Data $A$

$\text{cert}(q, T, A)$
Conjunctive Queries

- Formal counterpart of Select-Project-Join Queries in RA.
  \[ q(\vec{x}) \leftarrow \exists \vec{y}. \psi(\vec{x}, \vec{y}) \]

- \( \psi \) is a conjunction of atoms over constants and variables of the form:
  \[ A(t) \quad R(t, t') \]

- A Union of CQs (UCQ) is a disjunction of CQs, corresponding to a union of SPJs.
**DL-Lite\(A\)**

- Lightweight Description Logic tailored for accessing large data sources.
- Concepts and roles model set of objects and relationships among them.

\[ C \rightarrow A \mid \exists R \quad R \rightarrow P \mid P^- \]

- A **DL-Lite\(A\)** ontology \(\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle\) is composed of:
  - **TBox** \(\mathcal{T}\) Specifying constraints at the conceptual level.
    \[ C \sqsubseteq D \quad C \sqsubseteq \neg D \quad \text{(func} R) \]
    \[ R_1 \sqsubseteq R_2 \quad R_1 \sqsubseteq \neg R_2 \]
  - **ABox** \(\mathcal{A}\) Specifying the facts that hold in the domain.
    \[ A(b) \quad P(a,b) \]
The perfect reformulation *embeds* terminological information into $r_{q,T}$. 
Mock Ontology

PostGrad ⊑ Student
UnderGrad ⊑ Student
UnderGrad ⊑ ¬Postgrad
PartTime ⊑ Student

Professor ⊑ teaches ⊑ Course
Tutor ⊑ Professor
Advanced ⊑ Course

PartTime ⊑ hasTutor ⊑ Tutor
Tutor ⊑ hasTutor¬ ⊑ Tutor

PostGrad ⊑ Student
UnderGrad ⊑ Student
UnderGrad ⊑ ¬Postgrad
PartTime ⊑ Student

Giorgio Stefanoni (Oxford)
Query (1)

University Database:

- \textit{teaches}(craig, SWT)
- \textit{hasTutor}(peter, craig)

Query:

- \textit{Professor}(x) \leftarrow q_1(x)

\textit{cert}(q_1, \mathcal{T}, A) = \{craig\}

- In the database there is no information on Professors, how did the system retrieve the answer?
Query (2)

University Database:

\[ \text{teaches}(craig, \text{SWT}) \]
\[ \text{hasTutor}(peter, craig) \]

Query:

\[ q_2(x) \leftarrow \text{teaches}(x, y), \text{Advanced}(y), \text{hasTutor}(z, x) \]

\[ \text{cert}(q_2, \mathcal{T}, A) = \emptyset \]

- Why is \textit{craig} not an answer?
- Is \textit{SWT} an \textit{Advanced} course?
- Does \textit{craig} teach a course not listed in the database?
Outline

1. Foundations

2. Explaining Positive Answers

3. Explaining Negative Answers

4. Conclusions
Provide explanations of the following form:

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{hasTutor}(peter, craig)$</td>
<td>$craig$ tutors</td>
</tr>
<tr>
<td>$\exists \text{hasTutor} \sqsubseteq \text{Tutor}$</td>
<td>$craig$ is a Tutor</td>
</tr>
<tr>
<td>$\text{Tutor} \sqsubseteq \text{Professor}$</td>
<td>$craig$ is a Professor</td>
</tr>
</tbody>
</table>
Aim

Provide explanations of the following form:

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{hasTutor}(\text{peter, craig})$</td>
<td>$\text{craig$ tutors}$</td>
</tr>
<tr>
<td>$\exists \text{hasTutor} \sqsubseteq \text{Tutor}$</td>
<td>$\text{craig is a Tutor}$</td>
</tr>
<tr>
<td>$\text{Tutor} \sqsubseteq \text{Professor}$</td>
<td>$\text{craig is a Professor}$</td>
</tr>
</tbody>
</table>

**Strategy:** Gather information on how TBox axioms are used to generate the perfect reformulation.
PerfectRef \((q, \mathcal{T})\) in a (non-rigorous) Nutshell

- \(\{q\} \subseteq \text{PerfectRef} (q, \mathcal{T})\).
- For each \(r \in \text{PerfectRef} (q, \mathcal{T})\), we consider different cases:
  1. \(r(x) \leftarrow \text{Professor}(x)\) and \(\text{Tutor} \sqsubseteq \text{Professor} \in \mathcal{T}\). Then,
     \[r'(x) \leftarrow \text{Tutor}(x)\]
PerfectRef \((q, T)\) in a (non-rigorous) Nutshell

- \(\{q\} \subseteq \text{PerfectRef}(q, T)\).
- For each \(r \in \text{PerfectRef}(q, T)\), we consider different cases:
  1. \(r(x) \leftarrow \text{Professor}(x) \text{ and } \text{Tutor} \sqsubseteq \text{Professor} \in T\). Then,
     \[ r'(x) \leftarrow \text{Tutor}(x) \]
  2. \(r(x) \leftarrow \text{hasTutor}(x, y) \text{ and } \text{PartTime} \sqsubseteq \exists \text{hasTutor}\). Then,
     \[ r'(x) \leftarrow \text{PartTime}(x) \]
PerfectRef\((q, \mathcal{T})\) in a (non-rigorous) Nutshell

- \(\{q\} \subseteq \text{PerfectRef}\((q, \mathcal{T})\).
- For each \(r \in \text{PerfectRef}\((q, \mathcal{T})\), we consider different cases:
  1. \(r(x) \leftarrow \text{Professor}(x) \text{ and } \text{Tutor} \sqsubseteq \text{Professor} \in \mathcal{T}.\) Then,
     \[
     r'(x) \leftarrow \text{Tutor}(x)
     \]
  2. \(r(x) \leftarrow \text{hasTutor}(x, y) \text{ and } \text{PartTime} \sqsubseteq \exists \text{hasTutor}.\) Then,
     \[
     r'(x) \leftarrow \text{PartTime}(x)
     \]
  3. \(r(x) \leftarrow \text{Professor}(x) \text{ and } \exists \text{teaches} \sqsubseteq \text{Professor}.\) Then,
     \[
     r'(x) \leftarrow \text{teaches}(x, _)
     \]
Computing Positive Explanations

- Maintain a graph $G$ of rewritings.
  - $(r, r') \in G$ means that $r'$ has been generated from $r$.
  - Label $(r, r')$ with the axiom justifying the rewriting.
- Let $\pi$ be a match for $r \in \text{PerfectRef}(q_1, \mathcal{T})$ in $A$ witnessing $\text{craig}$.
- **IDEA:** Traverse backwards the trace of rewritings from $r$ until $q_1$ is reached. Suitably extend $\pi$ to be a match for intervening queries.
Example

\[ q_1(x) \leftarrow \text{Professor}(x) \]

\text{teaches}(\text{craig}, \text{SWT}) \quad \text{Database} \quad \text{hasTutor}(\text{peter}, \text{craig})
Example

\[ q_1(x) \leftarrow Professor(x) \]

\[ r_1(x) \leftarrow Tutor(x) \]

\textit{teaches}(craig, SWT) \quad \text{Database} \quad \textit{hasTutor}(peter, craig)
Example

\[ q_1(x) \leftarrow \text{Professor}(x) \]

\[ r_1(x) \leftarrow \text{Tutor}(x) \]

\[ r_2(x) \leftarrow \text{hasTutor}(y,x) \]

\( \pi \) matches \( x \) on \( \text{craig} \) and \( y \) on \( \text{peter} \).

\( \text{teaches(} \text{craig}, \text{SWT}\) \) 

\( \text{Database} \)

\( \text{hasTutor(} \text{peter}, \text{craig}\) \)
Example

\[ q_1(x) \leftarrow \text{Professor}(x) \]
\[ r_1(x) \leftarrow \text{Tutor}(x) \]
\[ r_2(x) \leftarrow \text{hasTutor}(peter, craig) \]

\[ \text{teaches}(craig, SWT) \]
\[ \text{Database} \]
\[ \text{hasTutor}(peter, craig) \]

\[ \pi \text{ matches } x \text{ on } craig \text{ and } y \text{ on } peter. \]
Example

\[ q_1(x) \leftarrow \text{Professor}(x) \]

\[ r_1(x) \leftarrow \text{Tutor}(\text{craig}) \]

\[ r_2(x) \leftarrow \text{hasTutor}(\text{peter}, \text{craig}) \]

\[ \text{teaches(\text{craig}, SWT)} \quad \text{Database} \quad \text{hasTutor(\text{peter}, \text{craig})} \]

\[ \pi \text{ matches } x \text{ on } \text{craig} \text{ and } y \text{ on } \text{peter}. \]
Algorithmic Solution

- Modify *PerfectRef* to maintain rewriting graph.
- At explanation time, use Dijkstra algorithm to find shortest path between generating rewriting and user query.
- Extend match on generating rewriting for intervening queries.
- Return shortest path and extended match.
Complexity

- Dijkstra runs in $O(|V|^2)$.
- In our case, the number of vertexes is the number of conjunctive queries in $\text{PerfectRef}(q, T)$.
- **Worst-case:** a CQ $q$ admits exponentially many rewritings w.r.t. $DL$-$\text{Lite}_A$ TBox $T$.
- Our explanation algorithm runs in exponential time w.r.t. the query.
- Data-complexity is still low.
Outline

1. Foundations
2. Explaining Positive Answers
3. Explaining Negative Answers
4. Conclusions
Query (2)

University Database:

\[ \text{teaches}(\text{craig}, \text{SWT}) \]
\[ \text{hasTutor}(\text{peter}, \text{craig}) \]

Query:

\[ q_2(x) \leftarrow \text{teaches}(x, y), \text{Advanced}(y), \text{hasTutor}(z, x) \]

\[ \text{cert}(q_2, T, A) = \emptyset \]
Method

- **Abductive Reasoning**: solutions are assertions to be added to the ontology leading the given tuple to be returned by the system.
- Solutions should be non-redundant: study minimality conditions!
Abductive Reasoning

- A form of non-sequitor argument, in which
  \[ \Gamma \not\models B \]
  but \( B \) is assumed to follow from the premises.
- Solutions are set of formulae \( \mathcal{E} \) such that
  \[ \Gamma \cup \mathcal{E} \models B \]
- Natural conditions over solutions:
  - **Consistency** \( \Gamma \cup \mathcal{E} \not\models \bot \)
  - **Minimality** \( \mathcal{E} \) is minimal wrt. some criterion.
Does there exist a (minimal) solution? (EXIST)

Does a formula $\alpha$ occur in all (minimal) solutions? (NEC)

Does a formula $\alpha$ occur in some (minimal) solution? (REL)

Is a set $\mathcal{E}$ of formulae a (minimal) solution? (REC)
We call $\mathcal{P} = \langle \mathcal{T}, \mathcal{A}, Q(\vec{x}), \vec{a} \rangle$ a QAP, where

1. $\langle \mathcal{T}, \mathcal{A} \rangle$ is a $DL$-$Lite_\mathcal{A}$ ontology.
2. $Q(\vec{x})$ is a Union of CQs.
3. $\vec{a}$ is a tuple of constants of matching arity.

A solution to $\mathcal{P}$ is an ABox $\mathcal{E}$ such that:

- $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{E} \rangle$ is consistent.
- $\vec{a} \in \text{cert}(q, \mathcal{T}, \mathcal{A} \cup \mathcal{E})$.

We denote with $\text{expl}(\mathcal{P})$ the set of all solutions to $\mathcal{P}$. 
Properties of QAPs

\[ \mathcal{P} = \langle \mathcal{T}, A, Q(\vec{x}), \vec{a} \rangle \]

- If \( \vec{a} \notin \text{cert}(q, \mathcal{T}, A) \), we call \( \vec{a} \) a negative answer to \( Q \) over the ontology.
- Negative answers exist only if the ontology is consistent.
- If the ontology is inconsistent, the QAP does have solutions.
- A solution \( E \) to QAP \( \mathcal{P} \) can introduce constants not occurring in the ABox \( A \).
We consider the four reasoning tasks over abductive problems under 3 different preference orders:

- no minimality condition,
- subset-minimality order denoted by $\subseteq$, and,
- minimum explanation size order denoted by $\leq$. 
### Query (2)

**University Database:**

- `teaches(craig, SWT)`
- `hasTutor(peter, craig)`

**Query:**

\[
q_2(x) \leftarrow teaches(x, y), \text{Advanced}(y), hasTutor(z, x)
\]

<table>
<thead>
<tr>
<th>ABox additions:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ <code>Advanced(SWT)</code></td>
<td></td>
</tr>
<tr>
<td>⊆ <code>teaches(craig, new : ALG), Advanced(new : ALG)</code></td>
<td></td>
</tr>
<tr>
<td>none <code>teaches(craig, new : TOC), hasTutor(new : Ben, craig), Advanced(new : TOC)</code></td>
<td></td>
</tr>
</tbody>
</table>
### Outline of Complexity Results

<table>
<thead>
<tr>
<th></th>
<th>$\leq$-EXIST</th>
<th>$\leq$-NEC</th>
<th>$\leq$-REL</th>
<th>$\leq$-REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>PTime</td>
<td>PTime</td>
<td>PTime</td>
<td>NP</td>
</tr>
<tr>
<td>$\leq$</td>
<td>PTime</td>
<td>$P^{NP}$</td>
<td>$P^{NP}$</td>
<td>DP</td>
</tr>
<tr>
<td>$\subseteq$</td>
<td>PTime</td>
<td>PTime</td>
<td>$\Sigma_2^P$</td>
<td>DP</td>
</tr>
</tbody>
</table>

- $\leq$-EXIST
- $\leq$-NEC
- $\leq$-REL
- $\leq$-REC

$\leq$ $\subseteq$ $\leq$ $\subseteq$
### Outline of Complexity Results

<table>
<thead>
<tr>
<th></th>
<th>$\leq$-EXIST</th>
<th>$\leq$-NEC</th>
<th>$\leq$-REL</th>
<th>$\leq$-REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>PTime</td>
<td>PTime</td>
<td>PTime</td>
<td>NP</td>
</tr>
<tr>
<td>$\leq$</td>
<td>PTime</td>
<td>$P_{\parallel}^{NP}$</td>
<td>$P_{\parallel}^{NP}$</td>
<td>DP</td>
</tr>
<tr>
<td>$\subseteq$</td>
<td>PTime</td>
<td>PTime</td>
<td>$\Sigma_2^P$</td>
<td>DP</td>
</tr>
</tbody>
</table>
Canonical Explanations

- If QAP $\mathcal{P} = \langle T, A, Q, \vec{a} \rangle$ has a solution, then there is a small solution.
- Finding a solution amounts to satisfy one of the CQs in $Q$.
- Satisfying a CQ does not require more than the number of terms contained in the query itself.
- Hence, one can find a solution by instantiating terms occurring in the query using a small number of new constants.
Complexity of $\subseteq$-EXIST

- A minimal solution to a QAP $\mathcal{P}$ exists iff $\mathcal{P}$ has a (general) solution.

Theorem

*For DL-Lite$_A$, EXIST is in PTime-complete.*

Upper bound intuition.

- Consider QAPs over CQs, general result for UCQs follows.
- Treat the body of the query as an ABox $\mathcal{E}$ and set $\mathcal{O} = \mathcal{O} \cup \mathcal{E}$.
- Replace each variable $x$ in $\mathcal{E}$ with a variable representative $a_x$.
- Use disjointness in $\mathcal{O}$ to enforce distinctness among constants. Thus, only variable representatives can be identified.
- Check satisfiability of the resulting ontology $\mathcal{O}$ without the UNA.
Complexity of $\subseteq$-NEC

- An assertion is $\subseteq$-necessary iff it is necessary.

Theorem

*For DL-Lite$_A$, NEC is PTime-complete.*

Upper bound intuition.

- We want to decide whether $A(a)$ is necessary for $P = \langle O, q, \bar{a} \rangle$.
- Check whether $A(a)$ is a consequence of $O$. In case return no.
- Create $P' = \langle O', q, \bar{a} \rangle$ by extending $O$ as follows:

$$
\mathcal{T} \cup \bar{A} \sqsubseteq \neg A \quad A \cup \{ \bar{A}(a) \}
$$

- Check that $P'$ does not admit solutions. If this is the case return yes.
Complexity of $\subseteq$-REL

Theorem

For DL-Lite$_A$, $\subseteq$-REL is $\Sigma^P_2$-complete.

Upper bound intuition.

- We want to decide whether $A(a)$ is $\subseteq$-relevant for $P = \langle T, A, q, \bar{a} \rangle$.
- Guess a derivation of one rewriting $r$ in $PerfetctRef(q, T)$.
- Guess a subset $E$ of the atoms of $r$.
- Guess an instantiation $E$ of the atoms in $E$.
- Check that $E$ is an explanation for $P$. (NP)
- Check that $E$ is minimal (coNP)
Complexity of $\subseteq$-REC

**Theorem**

For $DL$-$Lite_{A}$, $\subseteq$-REC is DP-complete.

**Upper bound intuition.**

- By definition of DP.
- A language $L$ is in DP if there are two languages $L_1$ and $L_2$, resp. in NP and coNP such that:

$$L = L_1 \cap L_2$$

- Thus

$$L_1 = \{ \langle P, E \rangle \mid E \in expl(P) \}$$

$$L_2 = \{ \langle P, E \rangle \mid \neg \exists E' \in expl(P) \text{ such that } E' \subset E \}$$

$$\subseteq$-REC = $L_1 \cap L_2$$
Outline

1. Foundations
2. Explaining Positive Answers
3. Explaining Negative Answers
4. Conclusions
Conclusions

- Provide an algorithmic solution to the problem of explaining positive answers.
- Contribute with a new formalization to the problem of explaining negative answers over ontologies as an abductive task.
- For $DL-Lite_A$, we study the complexity of reasoning over QAPs under minimality conditions.
Publications
